Access Introductory

Filtering by selection

February 1998

Filter by selection is a great boon, under-used in my humble opinion and again worth passing on to your users.

Suppose that you have a large table of data (viewed either as a table or via a form). If you move the cursor into a given field and press the 'Filter by Selection' button, you can effectively query the table to reveal only those records which have the value in the field that you selected.

So, given 8,000 Customer records, you can filter out those customers who live, for example, in Worcester. However it gets better because you can then select another field and, by clicking the button again, you effectively add the value in that field to the selection criteria. This makes it very easy to home in to the records that you want.

Access Intermediate

Concatenation in queries and reports

July 1997

Many people know that you can use the + operator to concatenate text strings. Thus:-

[FirstName]+[LastName]

in a query will join the two strings together. This unhelpfully yields names like:-
BrianJones

You can simply add in a literal space like this:-

[FirstName] + " " + [LastName]

which then gives:-

Brian Jones

However, you can also consider using the lesser known & operator. Amazingly:-

[FirstName] & " " & [LastName]

gives exactly the same result:-

Brian Jones

"Hello", you're thinking, "He's flipped. If it gives the same result, why bother to use it?"

The advantage of getting into the habit of using & is that it automatically converts all data types into text. This has no effect if the two fields are already text (as in the case above) but it does wonders with expressions like:-

[FirstName] & " " & [DateOfBirth]

which will produce:-

Brian 06/04/1967

rather than the:-

#Error

produced by the + operator when it tries to concatenate data from two disparate data types.

In other words, if you habitually use & instead of + you won't have to use type conversion functions.

You could even use it in a report like this:-

"Report generated " & Now() & " for " & [Customer]

Access programming

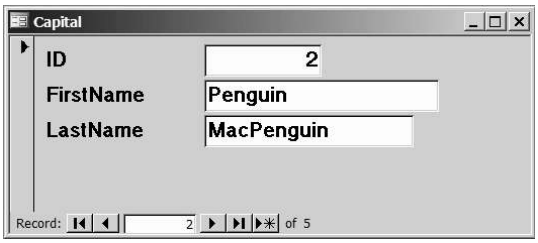
Capitalising names

April 1997

A reader requested a simple way to turn a name, say, 'fred smith' into 'Fred Smith' automatically as it is typed. (I realise that astute readers will already be thinking "Ah, what about the Mcs and the Macs and…." but let's solve the simple case first.)

The answer is that you have to use code, but only one line so it may just meet the requirement for simplicity.

Suppose you have a form called Capital which displays a field called FirstName in a text box called FirstName.



Switch to design, double click on the textbox in question, select the Event properties tab, click on the one called 'After Update', click on the ellipsis button which appears (three dots) and choose Code Builder. Between the two lines which appear:-

```
Private Sub FirstName_AfterUpdate()
End Sub
```

simply add a line so that it now reads as:-

```
Private Sub FirstName_AfterUpdate()
    Me!FirstName = StrConv(Me!FirstName, 3)
End Sub
```

'Me' means the current form, 'FirstName' is the name of the textbox, and 'StrConv' is a function (only available in Access 7.0 and above) which converts strings. The '3' tells the string conversion function to do the type of conversion that you asked for (capitalising the first letter of each word). So:-

```
Me!FirstName = StrConv(Me!FirstName, 3)
```

means "make the contents of the textbox called FirstName equal to the same as it is now, but with all of the first letters of the words capitalised".

This will convert:-

fred to Fred
sally to Sally
penguin penguinsson to Penguin Penguinsson
23 the larches to 23 The Larches
mcdonald to Mcdonald

etc.

I have always found that whatever product I use, eventually I reach a stage where I have to use code simply because it is impossible for the designers to put everything into the 'easy' set. On a happier note, once you do start to use code, your horizons expand considerably. You can, for example, begin to deal with unusually capitalised names like those mentioned above. If you expand the code for the Last Name field to:-

```
Private Sub LastName_AfterUpdate()
Dim Length As Integer

Me!LastName = StrConv(Me!LastName, 3)
If Left(Me!LastName, 3) = "Mac" Then
   Length = Len(Me!LastName)
```

```
   Me!LastName = Right(Me!LastName, Length - 3)
   Me!LastName = StrConv(Me!LastName, 3)
   Me!LastName = "Mac" + Me!LastName
End If

If Left(Me!LastName, 2) = "Mc" Then
   Length = Len(Me!LastName)
   Me!LastName = Right(Me!LastName, Length - 2)
   Me!LastName = StrConv(Me!LastName, 3)
   Me!LastName = "Mc" + Me!LastName
End If

End Sub
```

This will change macdonalds to MacDonalds, and mctavish to McTavish.

See the sample database.

Please note, I am not suggesting that this is perfect code. As usual in the sample code I give, there is no error trapping. In addition, there are some people who prefer the letter after Mac or Mc to be left in lowercase. This code is simply provided as an example of what can be done.

Database introduction

Talking telephone numbers

February 2000

A reader has inherited a database of 3,000 rows. All the telephone numbers are missing the leading 0 and the prospect of hand-editing them is not an enticing one.

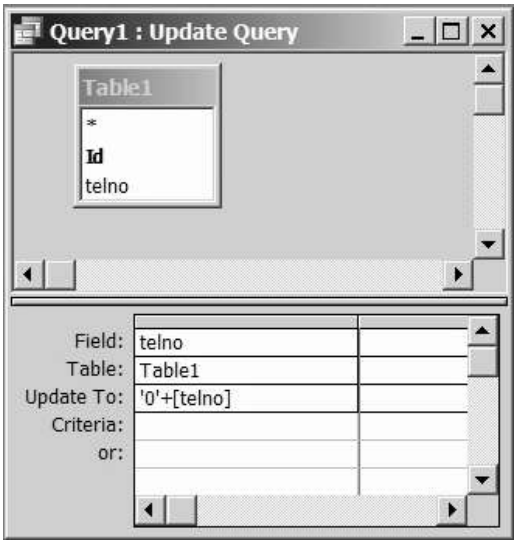
The first point to check is that the numbers are being stored in a text field not in a number field. This may sound odd at first but there are two good reasons why, in database terms, telephone numbers are not really numbers.

The first is that number fields typically suppress leading zeros, so 09230 is stripped down to 9230. This makes no sense at all for a phone number. Secondly, we chose the data type that is right for the data. The number data type allows you to perform mathematical operations of the data therein. Unless you are in the habit of calculating the average phone number of your friends, it isn't a good idea to store phone numbers in a number field; text is much better.

Assuming these are already in a text field, then an UPDATE query should do the trick. Given that the name of the field is, say, TelNo, then the SQL expression to use in the update query will be something like:

```
UPDATE Table1 SET Table1.telno = '0'+[telno];
```

Clearly you will have to substitute the appropriate table and field names. You can build this is raw SQL or if you are using Access you can use the query builder (see below).



This is clearly a specific answer to the reader's question, but it applies generally. You should never have to manually update lots of individual records. If you haven't come across update queries before they are well worth investigating in detail because they can do so much work for you, enabling you to make changes to all (or some) of the records in a table with a single command. Is it a bird? Is it a plane? No, it's Update Query!

Database intermediate

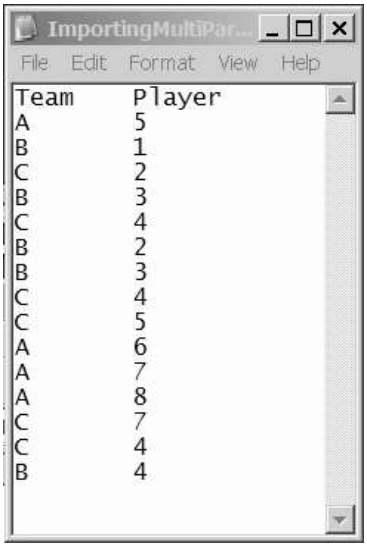
Importing multi-part keys

July 1994

A reader is having problems importing data from a text file. The data consisted of multiple columns, but the sticking point was the primary key. Most tables have one column as the primary key but it is not uncommon to have a key made up on two or more columns – a so-called multi-part (or joint) primary key.

Column 1 (Team) held the team name and column 2 (Player) the player's number. A table has been created in Access with a single multi-part key based on Team and Player. The table could be created fine but, when on trying to import the data, a "duplicate" record error occurred.

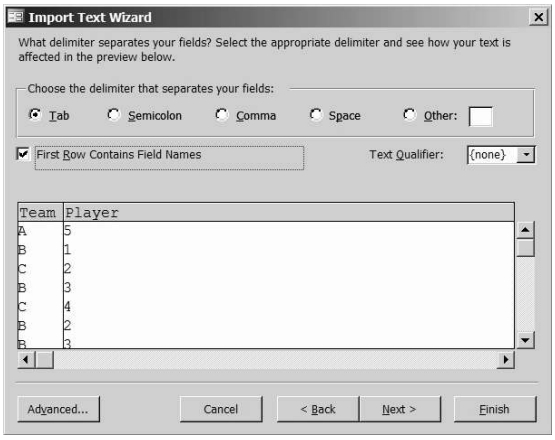
I sympathise with the problem and it is often difficult to spot duplicates in text files. For example, take a look at this text file:



Even though this is a very short file, spotting the duplicates is hard.

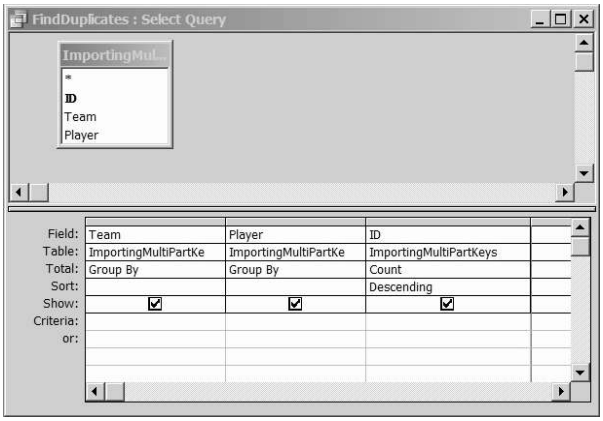
The answer is simply to import the data without applying a key, use a query to find the duplicates, remove them and then apply the key. This applies no matter which RDBMS you are using.

The sample file above is provided as ImportingMultiPartKeys.txt. You can import it into the database of your choice. In Access, open a new database and, from the menu, select File, Get External Data, Import – choose Text file as the type of file, navigate to the file and double click it. Choose delimited and, in the next dialog box, select Tab and First Row Contains Field Names.

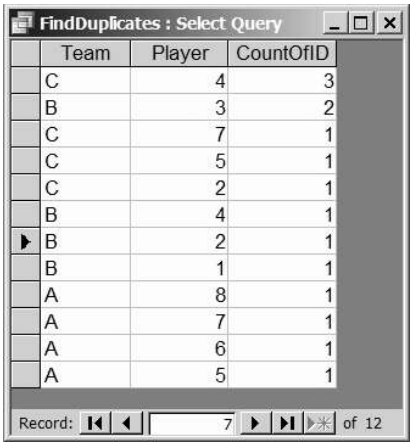


Accept the rest of the defaults and the data should appear in a new table; complete with a primary key that Access has added called ID

In the query design, drag Team, Player and ID down to the QBE grid. Then push the sigma button on the toolbar and set the Total row to Group By in the first two columns and Count in the third; also selecting Sort:Descending for the ID field.



Run the query and if there are any duplicates they will appear at the top of the table.



You can award yourself extra cool points if you thought of adding >1 as the criteria under ID. This will ensure that only the duplicates will appear.

You can now identify the duplicates and deal with them as you see fit. After that, you can apply the correct multi-part primary key.